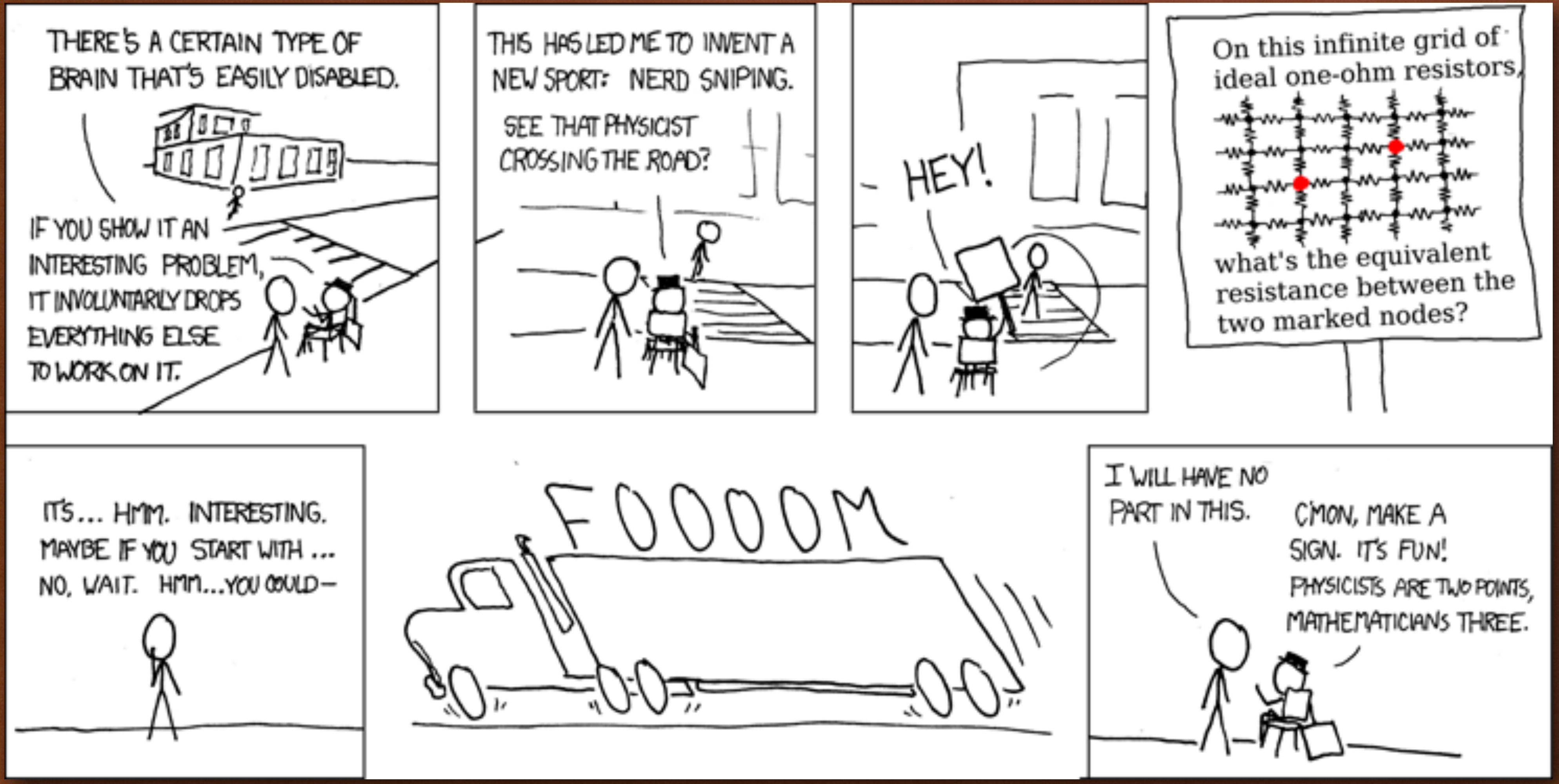
THE BUILDING BLOCKS: BOOLEAN LOGIC AND GATES



https://xkcd.com/356/

LEARNING OBJECTIVES

- Simple binary functions
  - one and two inputs with a single output
- Boolean values = binary values
  - Boolean logic
- Truth tables
- Transistors as switches
- Building boolean/binary functions with transistors
- Logic gates

HOW DO WE MANIPULATE NUMBERS IN CIRCUITS?

- We have seen how it is possible to encode properties about the world (including text, sound and images) into numbers
- We have seen how to encode numbers into binary, either integers or floating point
  - remember the "Why Binary" slide in Lecture 2
- How do we manipulate those numbers? i.e. perform operations on them?
- **We are going to build circuits to provide the operations we want to perform on those numbers**
  - **the presence or absence of a voltage on a wire will indicate 1 or 0**
  - **we will have many wires e.g. 8 to carry an 8-bit number and pass the values through a circuit which produces an 8-bit answer coming out on 8 wires**

STILL CHAPTER 4

SIMPLEST EXAMPLE - ONE INPUT FUNCTIONS

- All possible functions which take a value of either 0 or 1 and transform those values
- 1 input (A in this case) can take on 2^1 or 2 different values, so we have two rows in our table and 2^2 or 4 different functions
- These functions are known as the "zero", "identity", "not" and "one" functions
- The "zero", "identity" and "one" functions are trivial to implement in a circuit

A	Output
0	0
1	0

A	Output
0	0
1	1

A	Output
0	1
1	0

A	Output
0	1
1	1

TWO INPUT FUNCTIONS

- What do we get if we allow two inputs A and B?
- We have 2^2 rows and 2^4 or 16 different functions
- The coloured functions are special

A	B	Output
0	0	0
0	1	0
1	0	0
1	1	0

A	B	Output
0	0	1
0	1	0
1	0	0
1	1	0

A	B	Output
0	0	0
0	1	1
1	0	0
1	1	0

A	B	Output
0	0	1
0	1	1
1	0	0
1	1	0

A	B	Output
0	0	0
0	1	0
1	0	1
1	1	0

A	B	Output
0	0	1
0	1	0
1	0	1
1	1	0

A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0

A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0

A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1

A	B	Output
0	0	1
0	1	0
1	0	0
1	1	1

A	B	Output
0	0	0
0	1	1
1	0	0
1	1	1

A	B	Output
0	0	1
0	1	1
1	0	0
1	1	1

A	B	Output
0	0	0
0	1	0
1	0	1
1	1	1

A	B	Output
0	0	1
0	1	0
1	0	1
1	1	1

A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1

A	B	Output
0	0	1
0	1	1
1	0	1
1	1	1

ARE THEY REALLY NUMBERS?

- The two values 1 and 0 are completely arbitrary - how we interpret them depends on what we are doing
- **Boolean logic** is used for manipulating true/false expressions
- Binary 1/0 maps to true/false of Boolean logic
- Boolean expressions are true or false: $x \leq 35$, $a = 12$
- Boolean operators: $(0 \leq x)$ AND $(x \leq 35)$, $(a = 12)$ OR $(a = 13)$, NOT $(a = 12)$

$$(0 \leq x) \bullet (x \leq 35),\ (a = 12) + (a = 13),\ \sim(a = 12)$$

So AND can be represented as • (we will see that sometimes the • is not shown), OR can be represented as +, NOT can be represented as ~. NOT is also represented as a bar e.g. $\bar{a}$

TRUTH TABLE - AND

- **Truth tables** lay out true/false values for Boolean expressions, for each possible true/false input

FIGURE 4.14

Inputs: a	Inputs: b	Output a AND b (also written a . b)
False	False	False
False	True	False
True	False	False
True	True	True

or just **ab**

Truth table for the AND operation

TRUTH TABLE - OR

FIGURE 4.15

Inputs: a	Inputs: b	Output a OR b (also written a + b)
False	False	False
False	True	True
True	False	True
True	True	True

Truth table for the OR operation

TRUTH TABLE - NOT

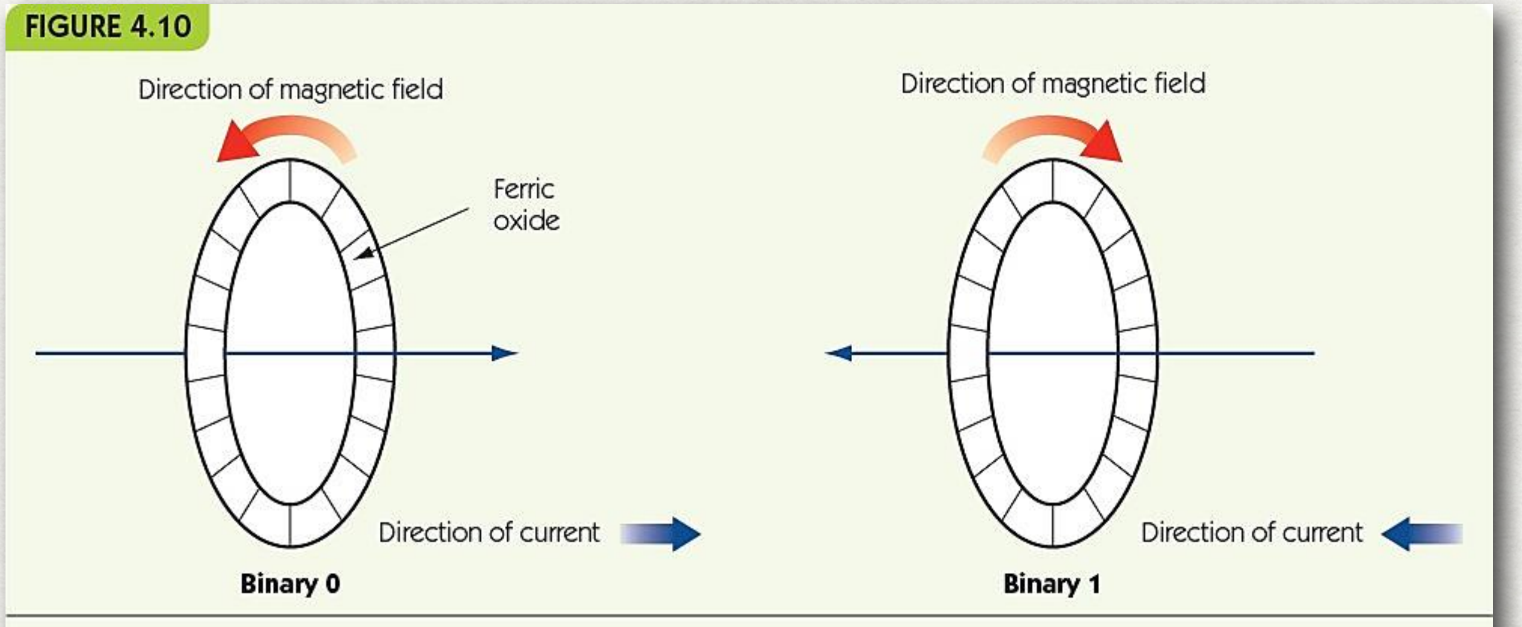
FIGURE 4.16

Inputs: a	**Output NOT a** (also written ā, or ~a)
False	True
True	False

Truth table for the NOT operation

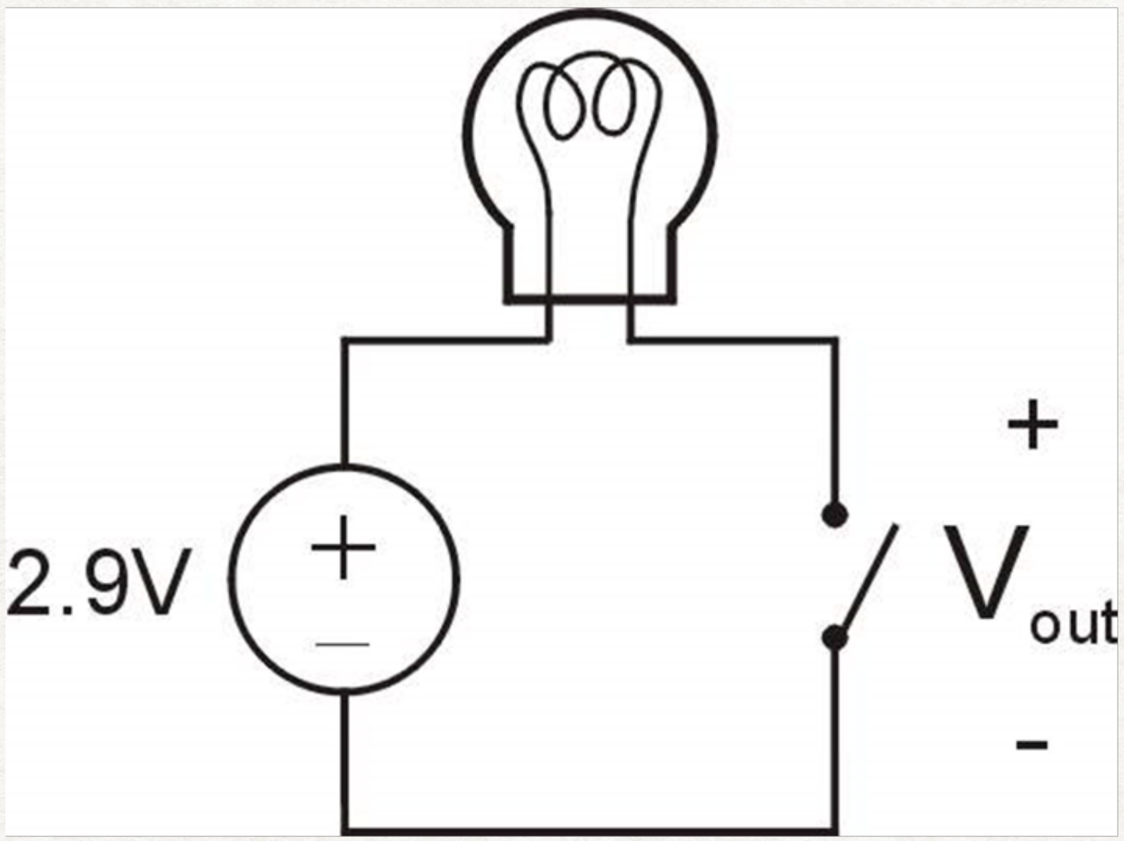
THE HARDWARE

- We need to be able to represent our two values and turn values on and off
- The ways of representing the two values have changed over time
- Computers use binary because "bistable" systems are reliable
    - Current on/off
    - Magnetic field left/right



Using magnetic cores to represent binary values

SIMPLE SWITCH



Switch open:

- No current through circuit
- Light is off

Switch closed:

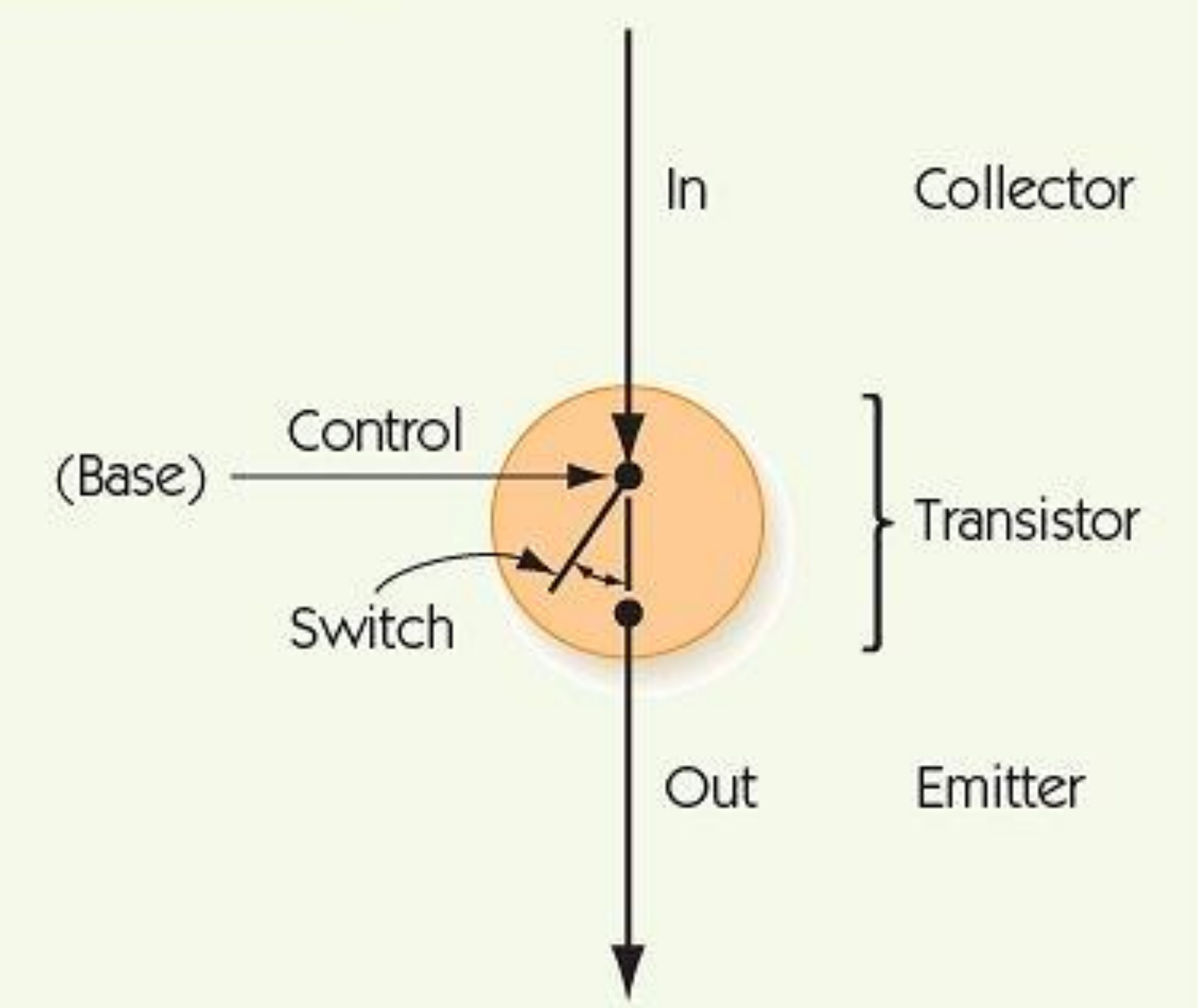
- Short circuit across switch
- Current flows
- Light is on

Switch-based circuits can easily represent two states: on/off, open/closed, voltage/no voltage.

TRANSISTORS

- **Transistors**
  - Solid-state switches
  - Change on/off when given power on control line
  - Extremely small (billions per chip)
  - Enable computers that work with **gigabytes** of data

FIGURE 4.12



Simplified model of a transistor

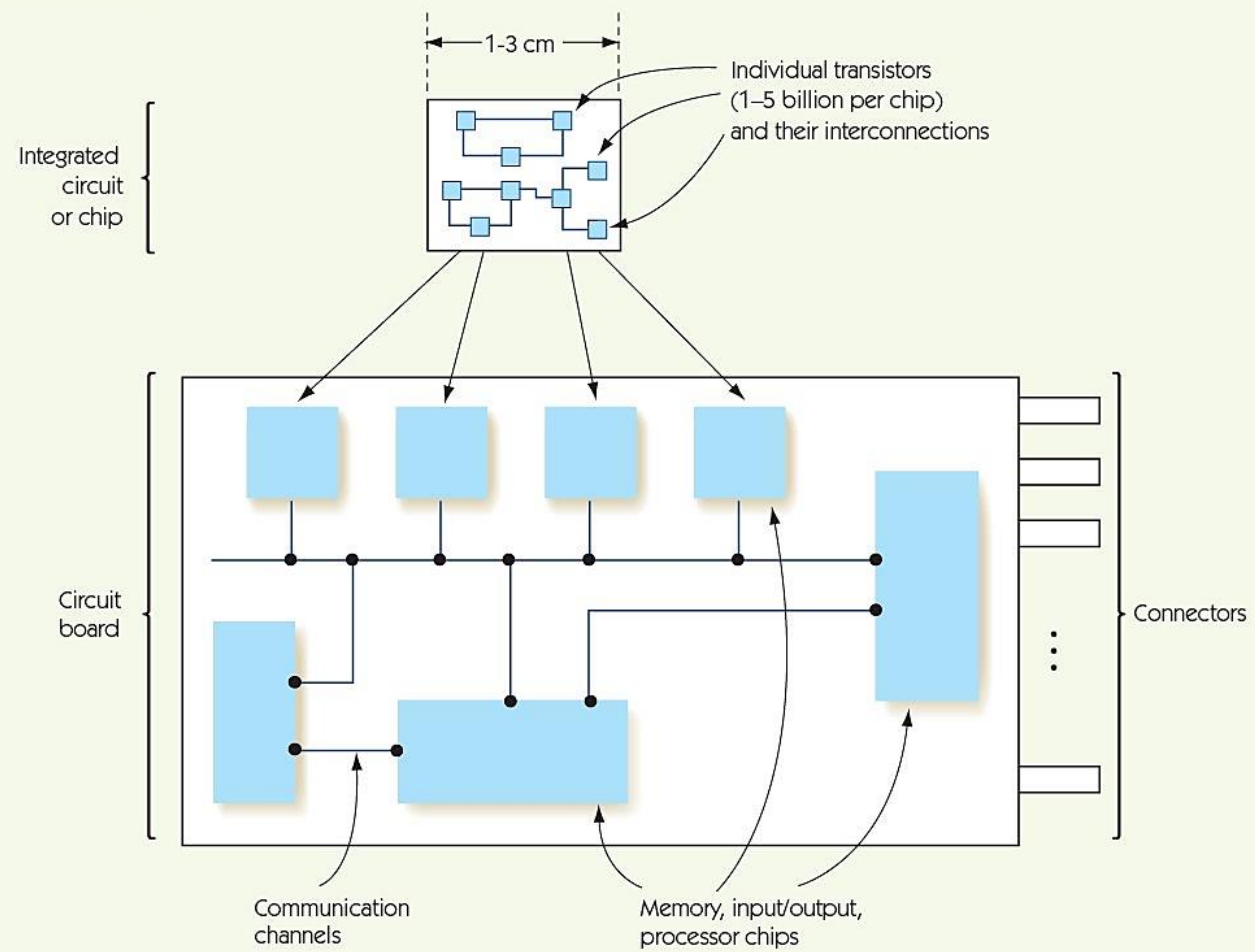
TRANSISTORS

- How do they work as switches? https://www.youtube.com/watch?v=stM8dgcY1CA start at 4:50 (not examinable but useful to have an overview of how transistors work)

- Great old video - https://www.youtube.com/watch?v=V9xUQWo4vN0 (not necessary for this course)

WHAT A COMPUTER LOOKS LIKE

FIGURE 4.11



Relationships among transistors, chips, and circuit boards

PROCESSORS AND TRANSISTORS

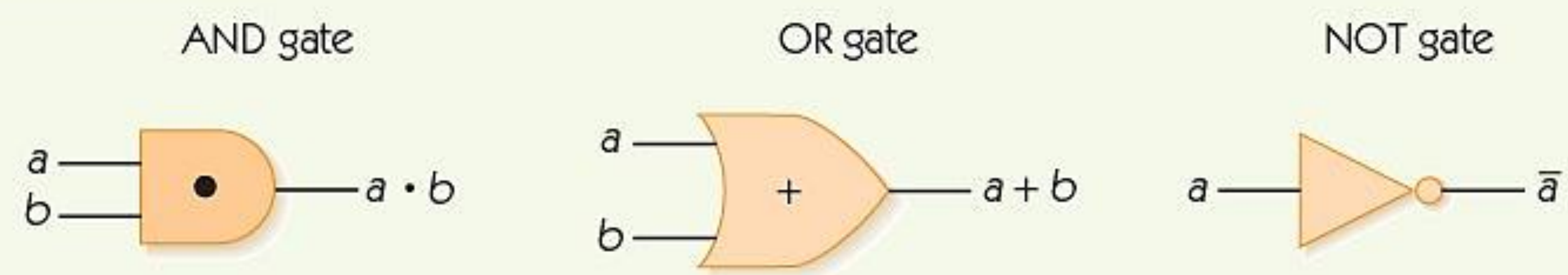
- Microprocessors contain millions of transistors
  - Intel® Xeon Phi™ coprocessor 5110P(2012): 5 billion
  - Spark M7 (2015): 10 billion
- Logically, each transistor acts as a switch
- Combined to implement logic functions
  - AND, OR, NOT - these are **functionally complete**, we can use them to create any logic function (see slide "Two Input Functions")
- Combined to build higher-level structures
  - Adder, multiplexer, decoder, register, …

GATES

- **Gate:** an electronic device that operates on inputs to produce outputs

- Each gate corresponds to a Boolean operator

FIGURE 4.17



a	b	a · b
0	0	0
0	1	0
1	0	0
1	1	1

a	b	a + b
0	0	0
0	1	1
1	0	1
1	1	1

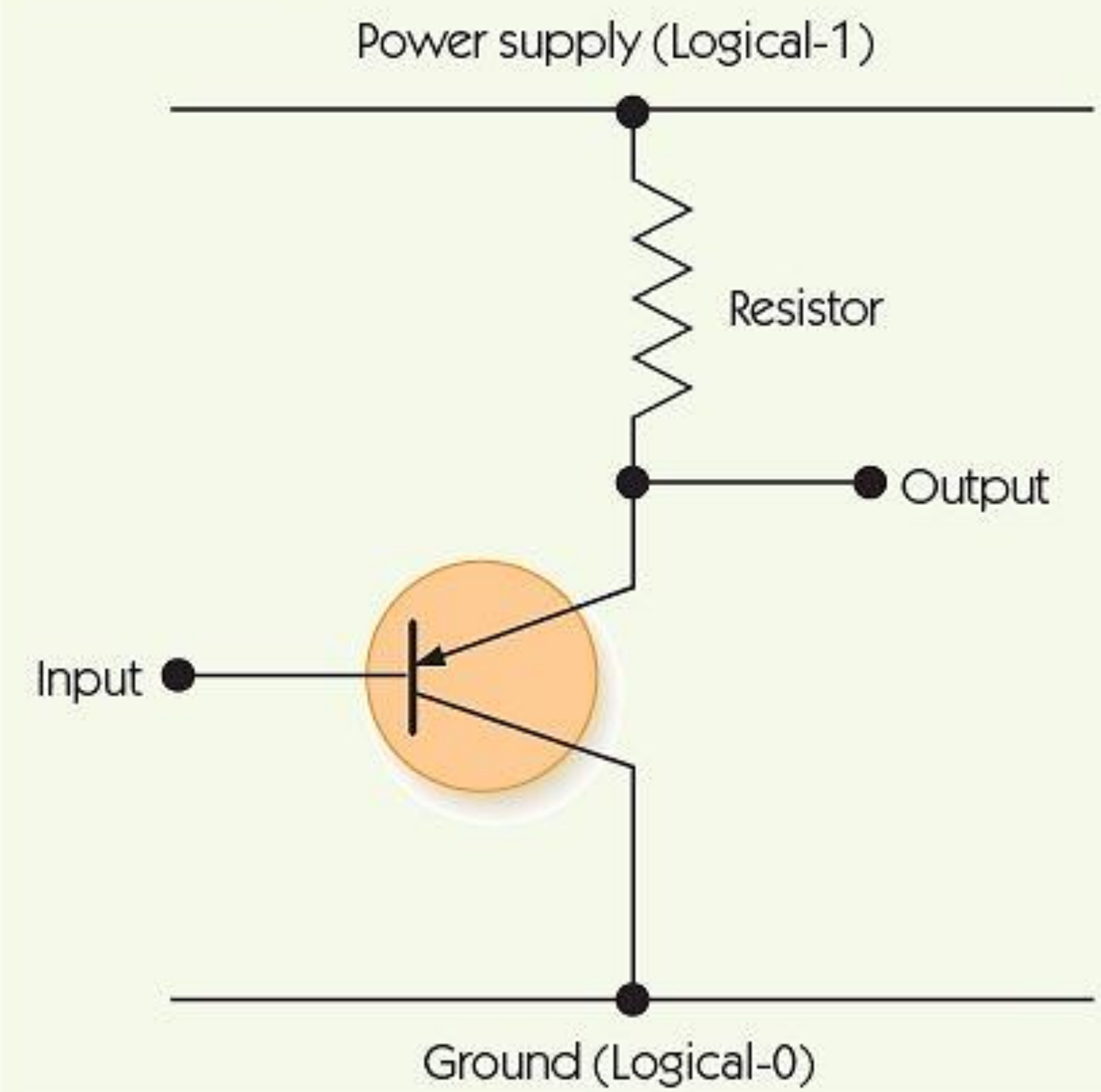
a	ā
0	1
1	0

The three basic gates and their symbols

NOT

- Gates are built from transistors
- NOT gate: 1 transistor
- AND gate: 3 transistors
- OR gate: 3 transistors
- NAND and NOR: 2 transistors
- Transistors can be in series or parallel

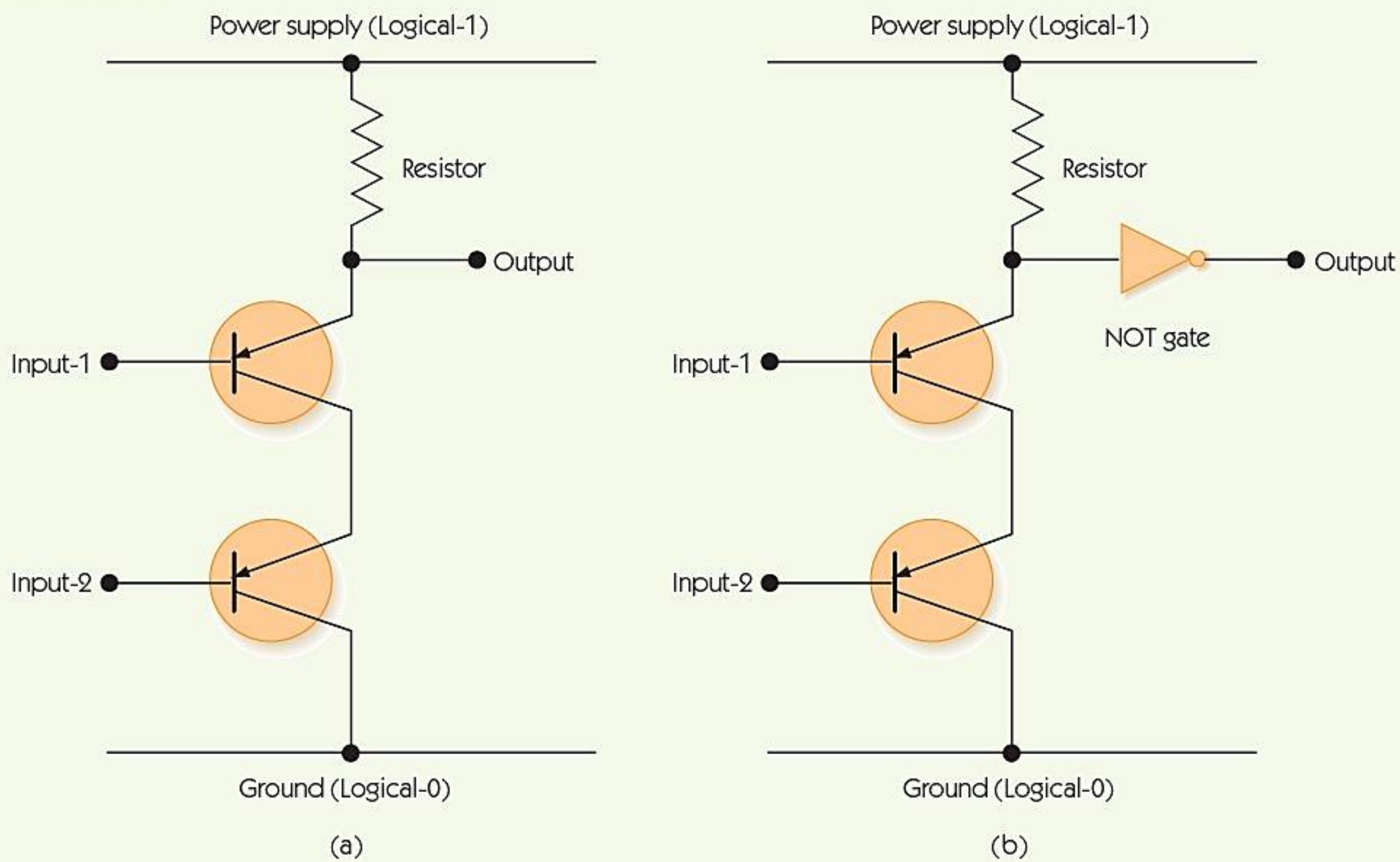
FIGURE 4.18



Construction of a NOT gate

NAND AND AND

FIGURE 4.19



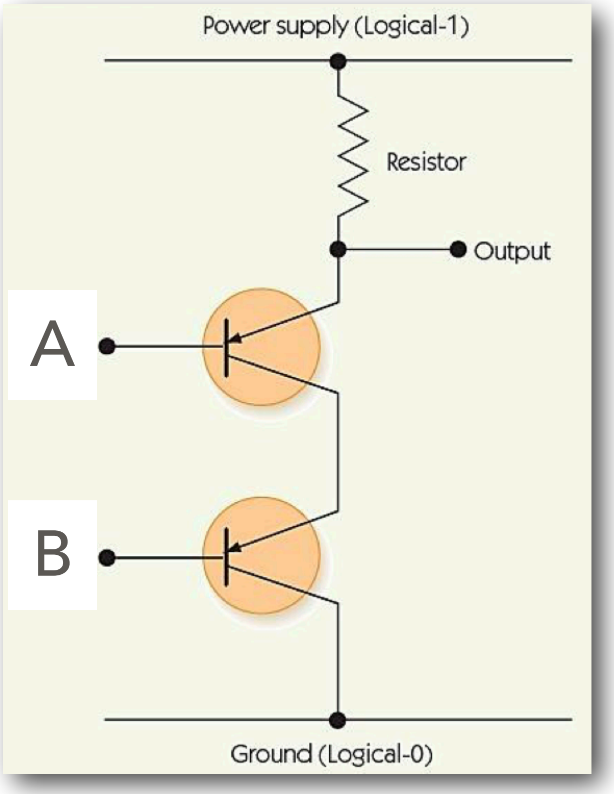
Construction of NAND and AND gates
(a) A two-transistor NAND gate
(b) A three-transistor AND gate

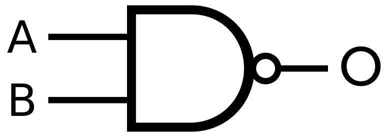
BUILD SOME OF THE TWO INPUT FUNCTIONS

out of the 16

A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0

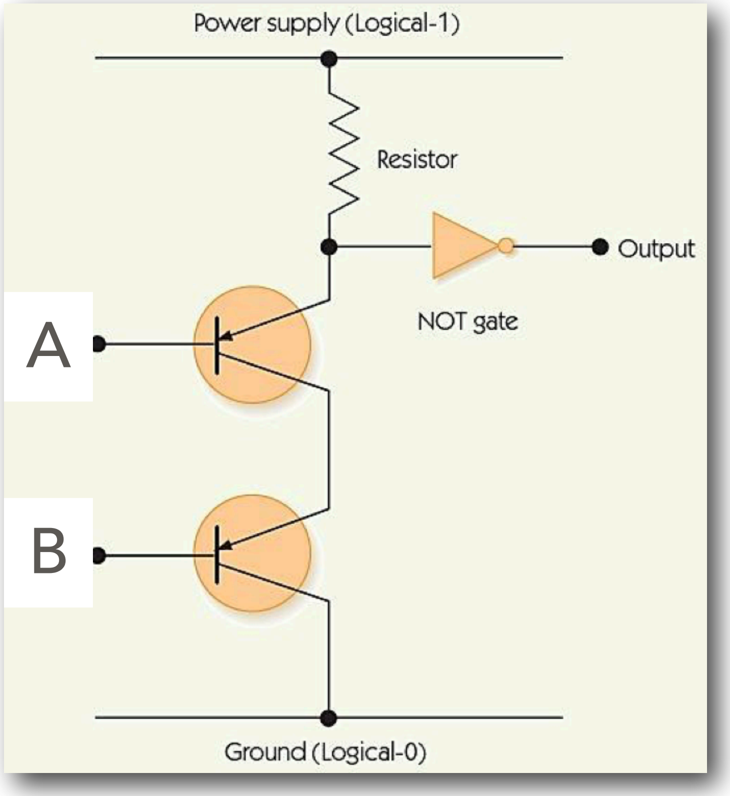
NAND

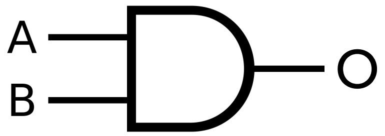




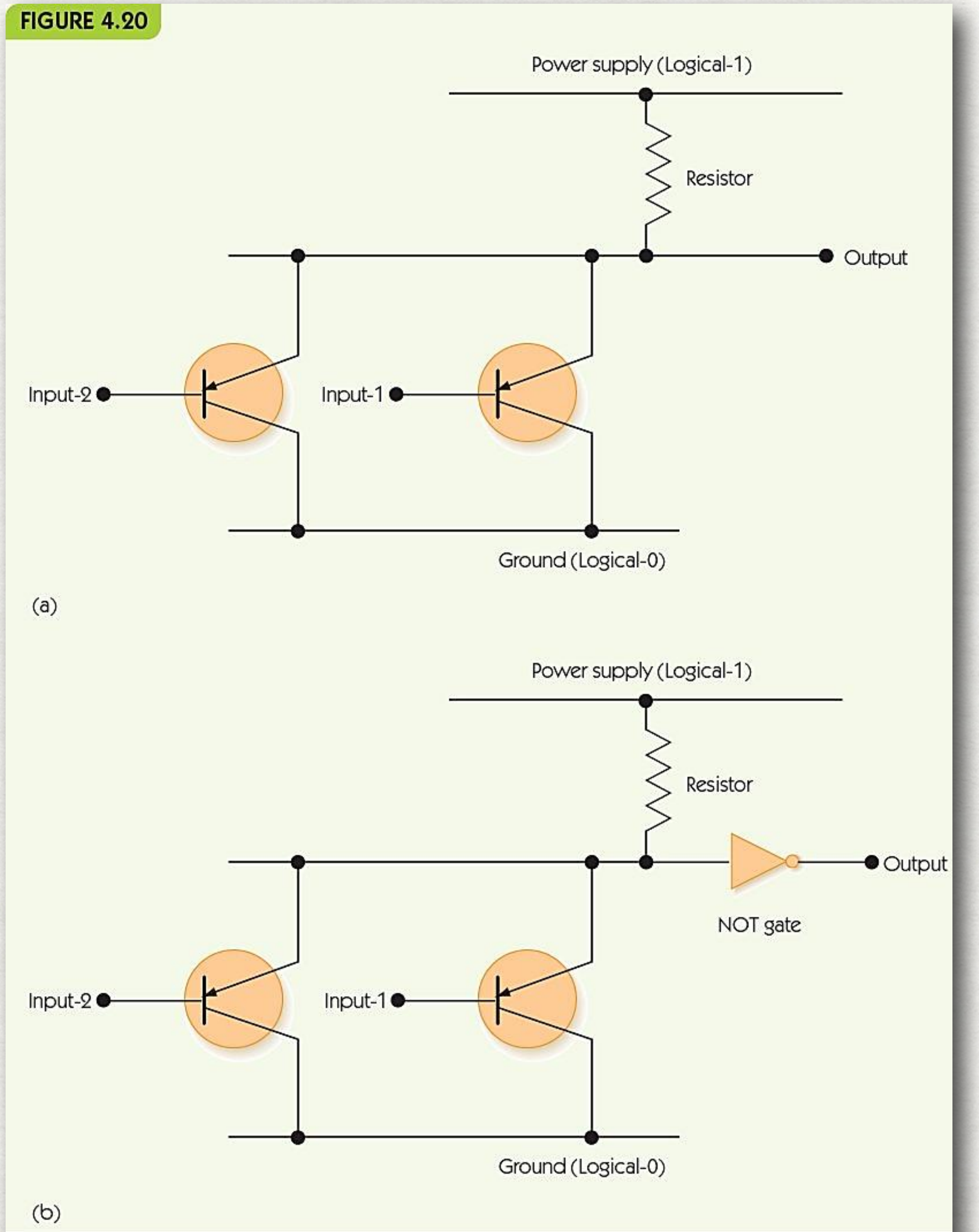
A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1

AND





NOR AND OR



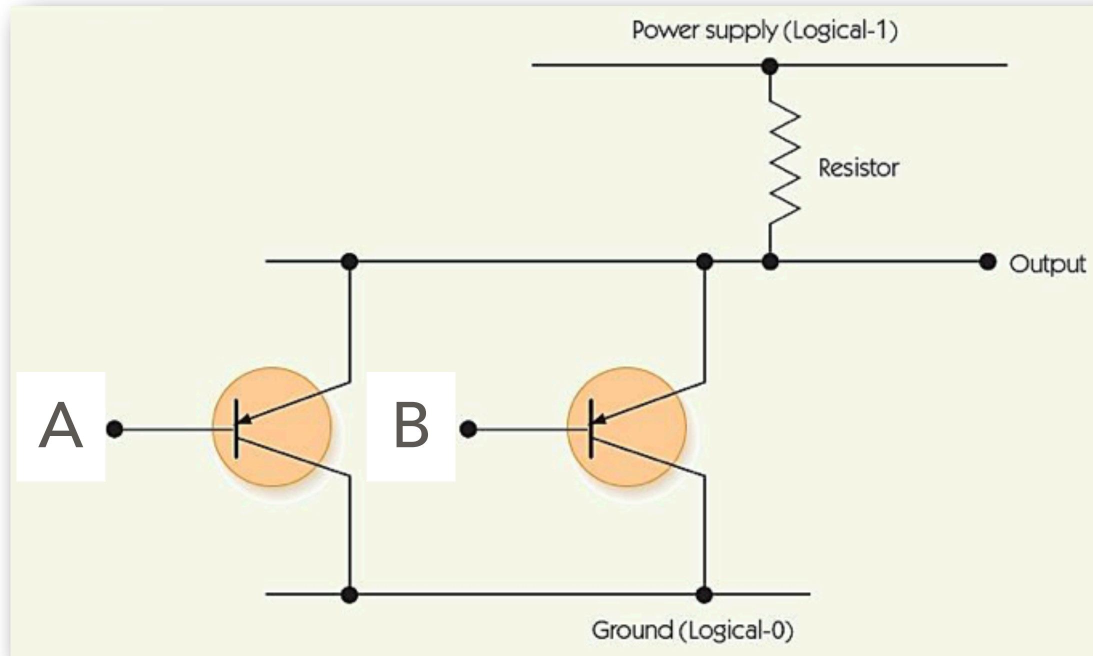
Construction of NOR and OR gates
(a) A two-transistor NOR gate
(b) A three-transistor OR gate

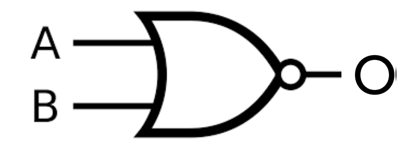
BUILD SOME OF THE TWO INPUT FUNCTIONS

out of the 16

A	B	Output
0	0	1
0	1	0
1	0	0
1	1	0

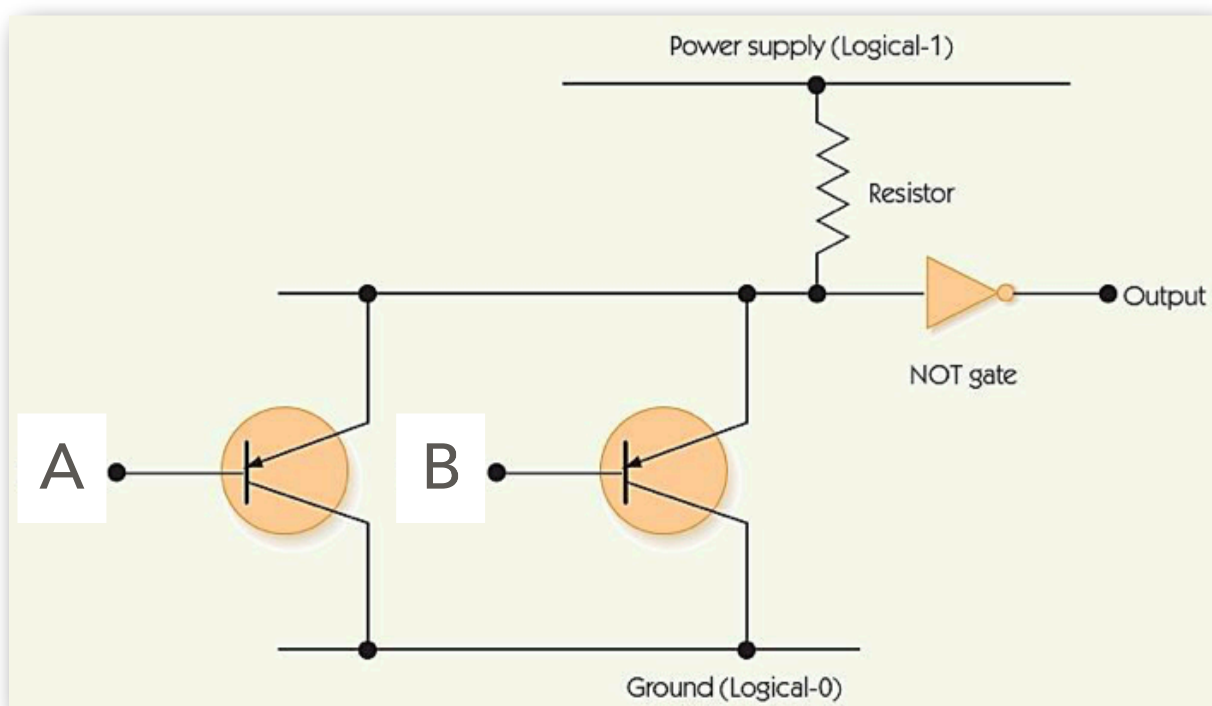
NOR

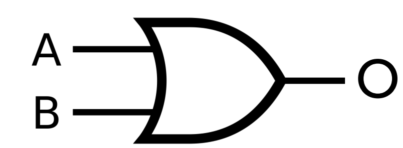




A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1

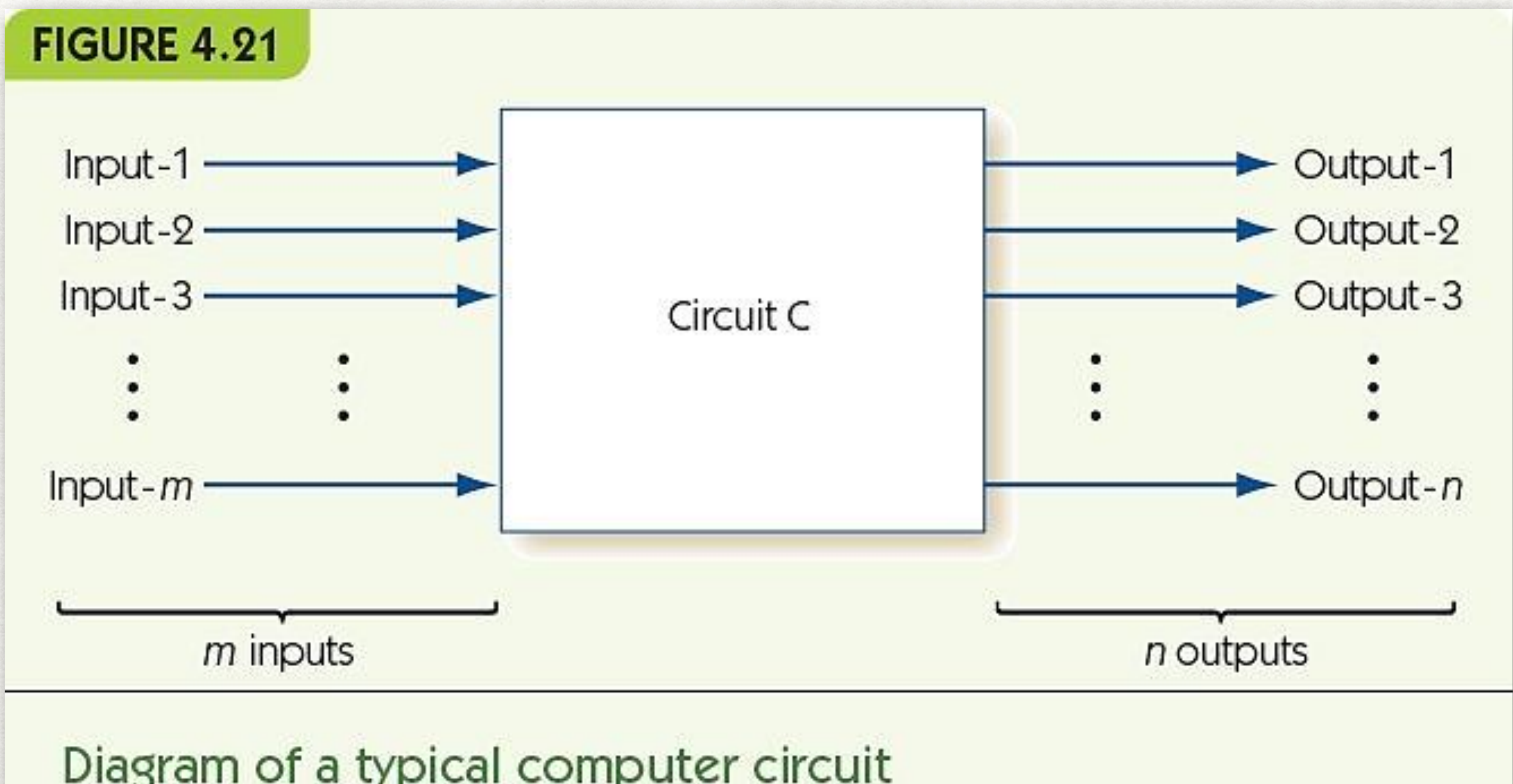
OR





COMBINATIONAL CIRCUIT

- **Circuit:** has input wires, contains gates connected by wires, and has output wires
- Outputs depend *only* on current inputs: no state



ABSTRACTION

- From now on we don’t draw the transistors in our circuits
- The basic elements will be the logic gates, NOT, AND and OR